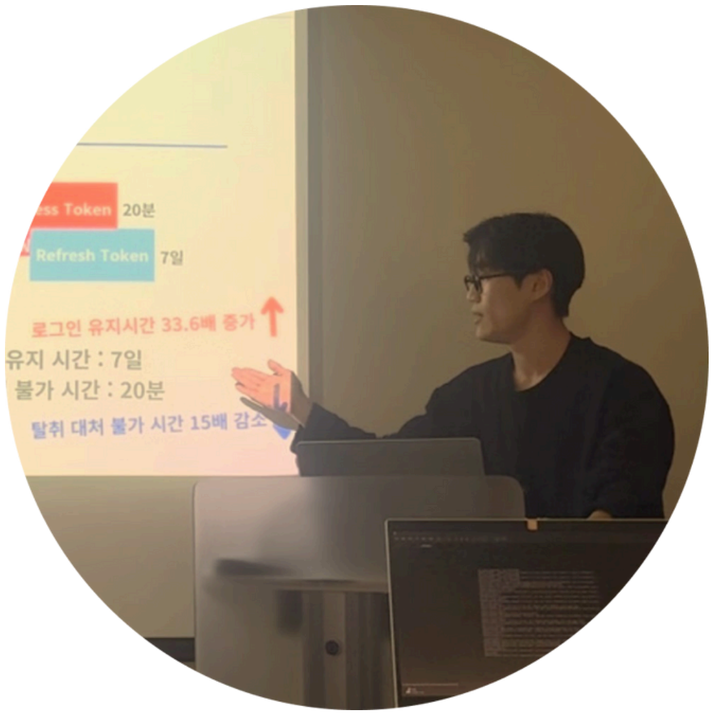
Portfolio

곽동현

Email	ehdgusjdbswl@gmail.com
Phone	010-2321-7659
GitHub	https://github.com/706com
Blog	https://develop-706.tistory.com/
Youtube	https://www.youtube.com/@딥담화



• 주요 기술 스택

- JAVA
- Spring Boot
- Spring Security
- MariaDB
- MySQL
- Redis

• 서브 기술 스택

- Spring Batch
- Docker
- kubernetes
- Kafka
- Jenkins

곽동현

모든 행동에 “WHY”를 담는, 백엔드 개발자입니다.

모든 결정과 행동에는 반드시 이유가 있어야 한다는 것을 핵심 가치로 삼고 있습니다.
구현 과정에 의미를 담으며 가치 있는 성과를 만들어내는 것을 목표로 합니다.

• 강점 키워드

- 끈기 / 집념
- 의사소통
- 호기심

• 학력 사항

2024.10 (수료) - 한화시스템 Beyond SW 캠프
- 백엔드 실무 6개월 과정
- 팀 프로젝트 5회 진행

2024.08 (졸업) - 안양 대학교
- 컴퓨터공학과 [4.17 / 4.5]
- 성적 우수 장학생 + KT 전액 장학생

2018.02 (졸업) - 성보 고등학교
- 이공계 학부

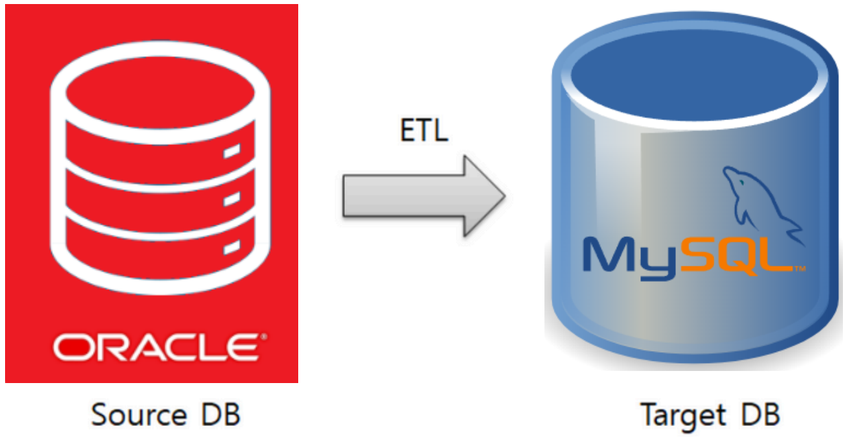
• 자격증

2024.08 - SQLD
2024.06 - 정보처리기사(필기)

Project &
Trouble Shooting

CDC Project

Oracle To MySQL
마이그레이션의 여정



Source DB　　Target DB

링크

- GitHub : https://github.com/DeepDamHwa/CDC_project

프로젝트 기간

2024.11 - 2025-01
(2개월)

프로젝트 인원

5명 (백엔드)

담당 기술 스택

- JAVA 17
- Spring Boot 3.x
- Spring Batch
- Docker
- Kafka
- Oracle DB
- MySQL
- Redis

담당 역할 & 개발 주요 상황

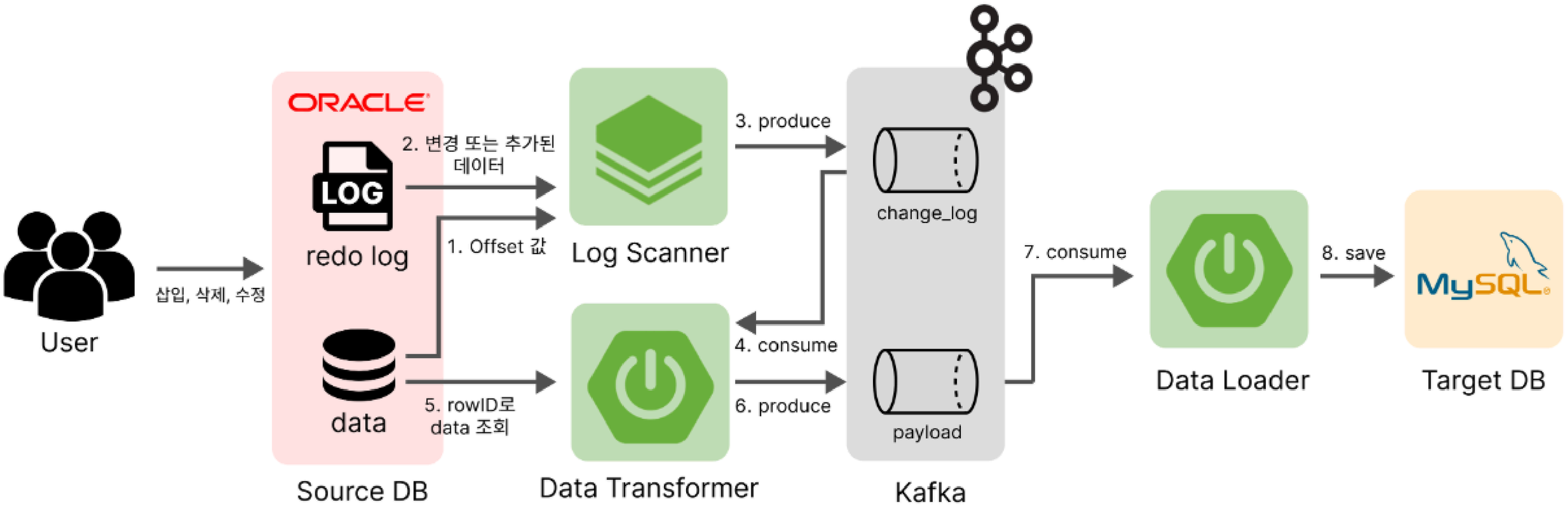
- LogMiner를 통한 트랜잭션 로그(Redo Log) 분석 및 관리

- LogMiner 를 활용하여 Redo로그에서 변화 데이터 추출
- 트랜잭션 격리 수준을 고려한 Offset 관리

- Batch 기반 CDC 작업

- 로그 데이터, 로그 파일등의 Offset을 활용한 데이터 추출
- Batch + Kafka 도입을 통한 비동기 처리
- 조회, 적재 작업 분리로 준실시간성 확보하도록 개선

시스템 아키텍처


User
삽입, 삭제, 수정
ORACLE
LOG
redo log
data
Source DB
2. 변경 또는 추가된 데이터
1. Offset 값
5. rowID로 data 조회
Log Scanner
Data Transformer
3. produce
4. consume
6. produce
change_log
payload
Kafka
7. consume
Data Loader
8. save
MySQL
Target DB

개선/문제 해결 사례 01

트랜잭션 격리 수준을 고려하지 못한 로그 수집 문제 해결

문제

- Redo 로그를 분석하고 데이터를 수집하는 과정에서 트랜잭션 격리 수준을 고려하지 않아 커밋되지 않은 데이터 또한 처리 되는 문제 발생
- 이에 따라, Source DB와 Target DB 간의 일관성 문제 발생

해결

- **Redo 로그에서 트랜잭션을 식별할 수 있는 고유 값(XIDUSN, XIDSLOT)을 활용하여 커밋이 완료된 데이터만 조회하도록 로직을 개선**

  -> 트랜잭션 단위 중 롤백이 일어났을 경우의 문제 가능성 제거

평가 (왜 이렇게 했고, 어떠한 결과가 있는가?)

- **커밋되지 않은 데이터를 읽고 처리할 경우**, 트랜잭션 실패 시 롤백이 반영되지 않으며, 이는 **데이터의 정합성 및 신뢰성 저하**로 이어질 수 있음
- **안정성과 일관성을 유지**하기 위해 커밋 완료된 로그만 Target 시스템에 반영하도록 설계

+ 추가 : OracleDB는 auto commit 기능이 없으므로, 수동 데이터 변경 시 반드시 commit 쿼리문을 실행해야 한다는 점을 확인

개선/문제 해결 사례 02

Source DB의 변화 데이터 누락 방지를 위한 로그 파일 / Offset 관리 문제 해결

문제

- LogMiner는 제공 받은 로그 파일 기준의 데이터만 바라보기 때문에, 로그 파일이 가득차서 변경되는 시점에서 일부 누락 데이터 발생 및 Offset 일관성 문제 발생

해결

- **트랜잭션 로그 데이터 중, RS_ID 컬럼 값을 활용해 어디까지 읽었는지 Offset 적용**

  -> 고유적인 논리적 주소를 의미하는 식별자(RS_ID)를 Offset 으로 활용하고 관리하여 로그 파일 변동 시 에도 순차적 데이터 일관성 유지

- **Redo 로그 파일 번호별 Offset을 추가 관리**

  -> 이전 Batch 작업에서 처리한 로그 파일부터 현재 파일 까지 모두 조회하는 로직 개선을 통해 데이터 누락 방지

평가 (왜 이렇게 했고, 어떠한 결과가 있는가?)

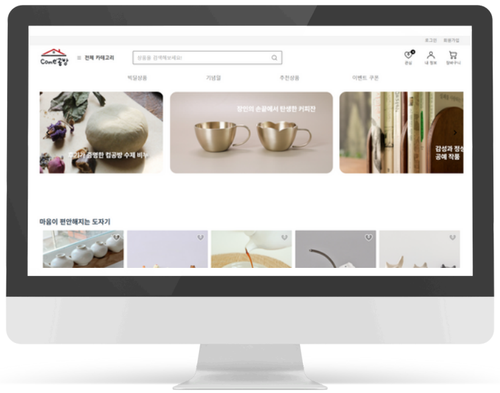
- 트랜잭션 로그는 고유 데이터이므로, 임의로 idx 컬럼을 추가하는 행위는 불가하여, 로그 파일 내에서 항상 오름차순으로 보장되는 RS_ID컬럼 활용
- 이를 PK와 유사하게 활용하여 별도로 테이블에 저장하고, 매 Batch 실행 시 해당 값을 통해 이후의 데이터를 읽도록 설정

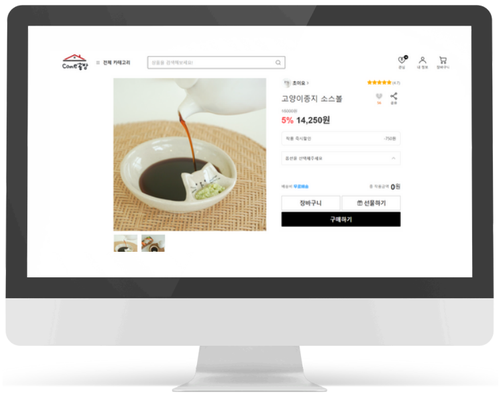
- Redo 로그 파일은 01 -> 02 -> 03 -> 01 과 같이 순환하며 파일을 작성하므로 로그 파일 추적에 대한 추가 관리가 필요

-> 이를 통해 매 Batch 작업때마다 누락되는 데이터 없이 Target System에 변화 데이터 적재 가능

Come’공방

Web 기반 공방 상품 판매
E-commerce 프로젝트





링크

- Site : https://comegongbangs.kro.kr/
- GitHub : https://github.com/706com/ComeGongBang

프로젝트 기간

2024.08 - 2024-11
(2개월)

프로젝트 인원

5명 (Full Stack)

담당 기술 스택

- JAVA 17
- Spring Boot 3.x
- Spring Security
- Spring Data JPA
- Docker
- DockerHub
- Kubernetes
- Jenkins
- MariaDB
- Redis
- S3

담당 역할 & 개발 주요 상황

- 회원 기능

- Kubernetes 분산 서버 세션 유지를 위해 스티키 세션 서비스 도입
- Redis Caching & TTL 을 활용하여 회원 토큰 기능 최적화

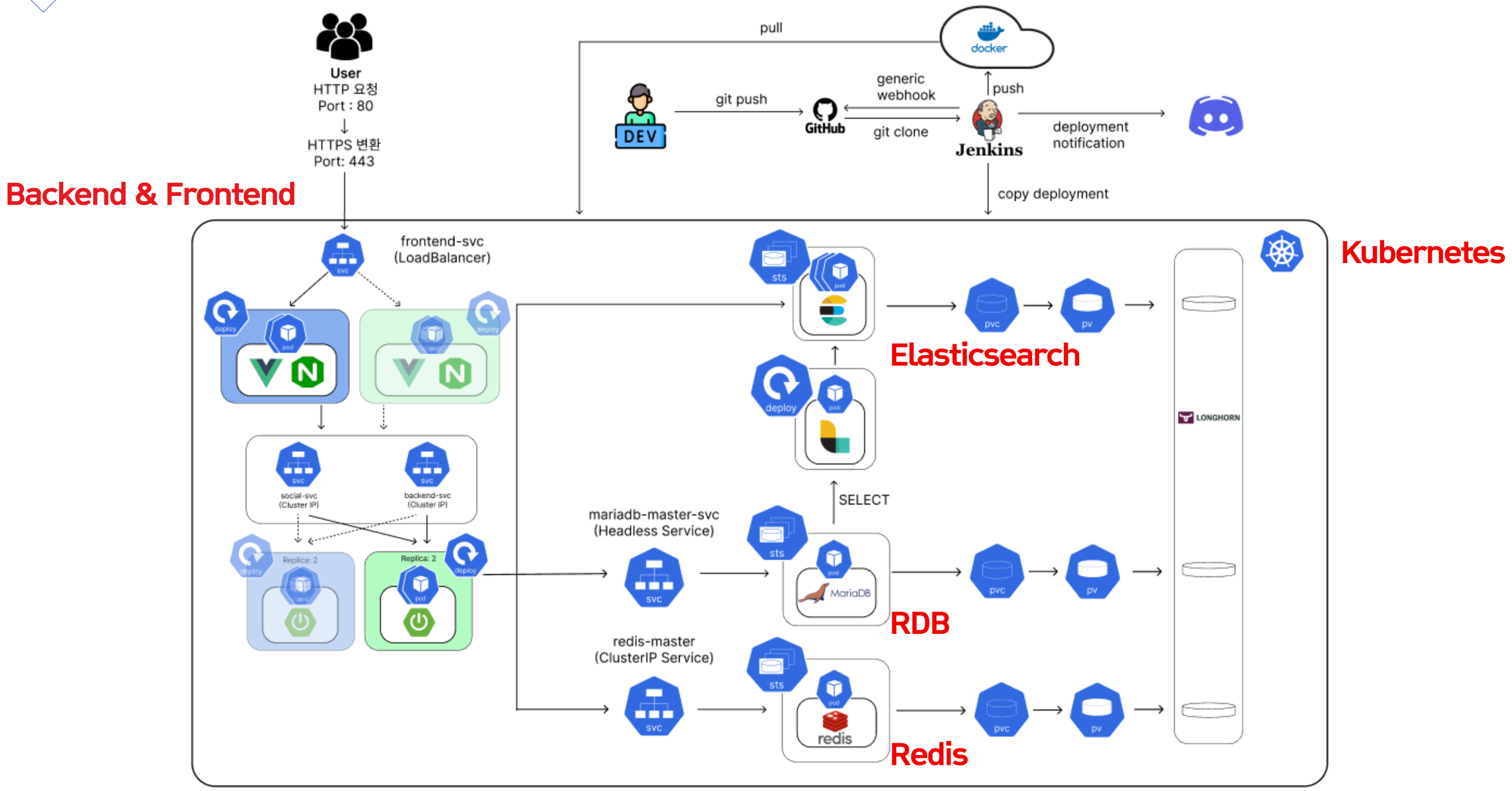
- 쿠폰 기능

- 대기열 시스템을 도입하여 대규모 트래픽 + 동시성 문제 해결

- CI/CD

- Kubernetes 환경 위에서 블루/그린 무중단 서비스 구성
- Jenkins + DockerHub를 활용한 CI/CD 구축

시스템 아키텍처


User
HTTP 요청
Port : 80
HTTPS 변환
Port: 443
Backend & Frontend
pull
docker
push
DEV
git push
GitHub
generic
webhook
git clone
Jenkins
deployment
notification
copy deployment
Kubernetes
frontend-svc
(LoadBalancer)
social-svc
(Cluster IP)
backend-svc
(Cluster IP)
Replica: 2
Replica: 2
SELECT
mariadb-master-svc
(Headless Service)
redis-master
(ClusterIP Service)
Elasticsearch
RDB
Redis
LONGHORN

개선/문제 해결 사례 01

분산 서버 OAuth2 인증 실패와 Kubernetes 스티키 세션 서비스 도입

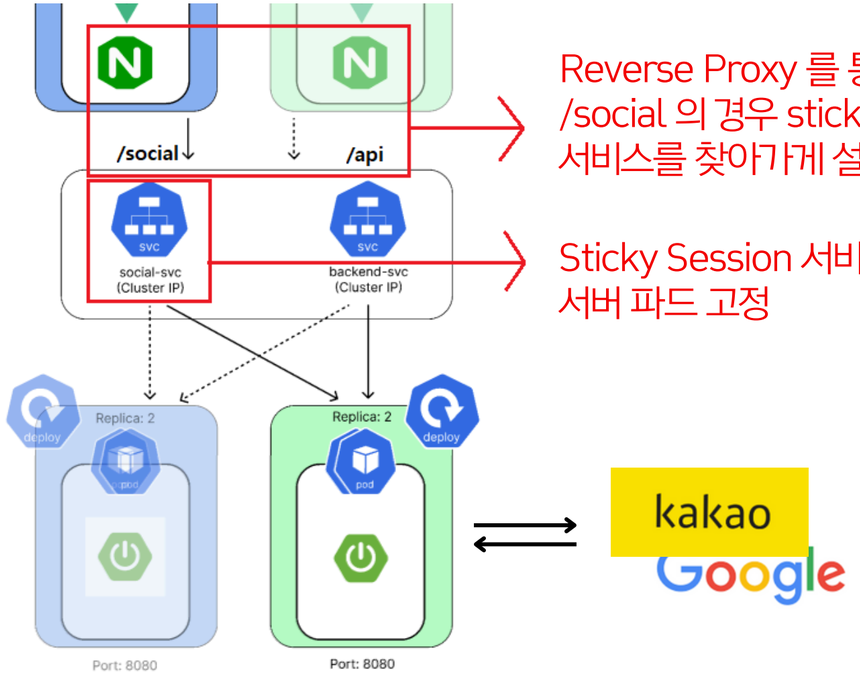
문제

- Kubernetes 위에서 동작하는 **분산 서버**에 의해 **OAuth2 소셜 로그인 과정이 실패**하는 문제 발생
- 소셜 인증 과정에서 인가코드 획득 -> 액세스 토큰 발급 단계에서 세션 정보가 끊기는 문제 지점 확인

해결

- **Kubernetes에서 OAuth2 소셜 인증을 위한 스티키 세션 서비스 도입**
  -> 인가 코드와 세션 정보를 통한 토큰 발급의 유기적 흐름 보장 성공

- **Nginx Reverse Proxy 설정**
  -> **/social** 의 형태로 URI 요청이 들어올 시
  스티키 세션 서비스로 찾아가는 Reverse Proxy 설정 추가



평가 (왜 이렇게 했고, 어떠한 결과가 있는가?)

- 해당 개발 단계에서는 Redis 도입 계획이 없었기 때문에, 구현의 간소화와 운영 비용 절감을 고려해 Redis 클러스터 방식 대신 Sticky Session 방식을 선택
- Refresh Token을 이미 RDB로 관리하고 있었기 때문에, 추가적인 **DB 부하를 방지하기 위해 토큰 방식을 채택 X**
- Sticky Session 방식 도입으로 로드 밸런싱에 제한이 있었지만, **SessionAffinity의 유효 시간을 1시간으로 설정**하여 이 문제를 해결

개선/문제 해결 사례 02

선착순 쿠폰 기능 동시성 문제 비관락 도입

문제

- 수량이 한 개 남은 상황 여러명이 동시에 쿠폰 발급을 시도할 시, 동시성 문제로 인해 다수의 요청이 충돌하여 정상적으로 쿠폰 발급이 되지 않는 상황 발생 (70명만 발급이 되거나, 120명이 발급되는 상황)

해결

- **비관적 락(Pessimistic Lock) 적용**

  -> 직접 DB에 락을 걸기 때문에 안정적인 동시성 제어 보장

Label	Throughput	Average	Error %
로그인	24.5/sec	991	0.00%
쿠폰 발급	27.2/sec	75	13.67%
TOTAL	48.9/sec	533	6.83%

TPS: 27/sec
응답 시간: 75ms
ERROR: 13.67%

비관적 락 적용 ↓

Label	Throughput	Average	Error %
로그인	24.1/sec	6890	0.00%
쿠폰 발급	24.7/sec	170	0.00%
TOTAL	48.1/sec	3530	0.00%

TPS: 25/sec
응답 시간: 170ms
ERROR: 0%

평가 (왜 이렇게 했고, 어떠한 결과가 있는가?)

- **비관적 락을 도입한 이유**

  -> Syncronized: 멀티 프로세스 환경에는 대처 불가

  -> 낙관적 락: 선착순 쿠폰 특성상, 순간적으로 트래픽이 몰리는 상황이 빈번하여, 오히려 취약할 수 있다고 판단

- 서로 다른 **1000명이 쿠폰 발급을 동시에 시도**했을 때 **발급이 실패할 확률 13.67% -> 0% 성능 개선**

부하 테스트 도구: Apache Jmeter

개선/문제 해결 사례 03

Redis 대기열 도입으로 선착순 쿠폰 기능 비관락 적용시 성능 저하 + 서버 다운 문제 해결

문제

- 비관적 락을 적용할 시, **요청 처리 속도가 급격하게 저하**되는 문제 발생
- 트래픽 급증 시 이를 DB 레벨에서만 처리하다 보니 **DB 병목 현상이 심해지며, 서버가 버티지 못하고 다운**되는 현상 발생
- 분산 환경 동시성: 서로 다른 서버에서 자원에 **동시 접근할 경우 지속적 경합** 발생

해결

- **분산락 적용을 통한 대기열 시스템 도입**

  -> 모든 요청을 한번에 처리하는 게 아닌 서버가 처리할 수 있을 만큼만 **순차적으로 처리**

  -> **싱글 스레드 구조**로 **순서대로 실행됨을 보장**하기 때문에 효과적인 대기 시스템 구현

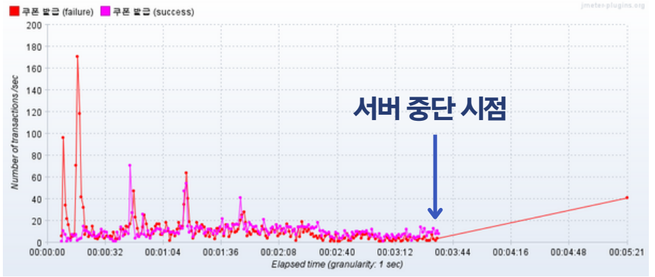
- **Redis Sorted Set 자료구조 선정**

  -> Sorted Set은 **Score를 기준으로 자동 정렬**되며 요청을 **시간순으로 Score를 부여**하고 저장 방식 채택

평가 (왜 이렇게 했고, 어떠한 결과가 있는가?)

- 수많은 트래픽이 몰릴 시 **서버가 죽는 현상** 발생 문제를 **대기열 도입 후 해결**

Redis 적용 후 성능 개선 수치



BEFORE

1초 1000명 응답 시간 지표

Error%

Label	# Samples	Throughput	Average	Error %
쿠폰 발급	9000	167.1/sec	10971	0.00%
TOTAL	9000	167.1/sec	10971	0.00%

AFTER

1초 3000명

0.00%

부하 테스트 도구: Apache Jmeter

Contact me

Email. ehdgusjdbswl@gmail.com
Phone. 010-2321-7659